Annotating the dynamic

Frank Sauerburger

Type Annotation for DataFrames

EuroSciPy, Krakow, 2025-08-19

Why?

- Function has type annotation
- **Does column article_title exist?**

- Hard to write
- Hard to review
- Hard to maintain

Breaks on Saturday at 3 am

Drop article with null titles
Frank Sauerburger authored 3 weeks ago

cfpworker/src/db/dwh_queries.py

cfpworker/src/suggestion_logic/filters/article_filters.py

```
1   1     import pandas as pd
2   2     from sniply.src.logs import Profiler
3   3
4   4
5   5     def article_based_filtering(
6   6         article_data: pd.DataFrame, publication_age_threshold: int = 5, profiler: Profiler = Profiler()
7   7     ):
8   8         """article_filtering_pre_author_selection Filter on the articles to apply before finding the author informatio
27  27        """
28  28        if len(article_data) == 0:
29  29            profiler.warning("No articles to filter", extra={"checkpoint": 107001})
30  30            return article_data
31      -     article_data = article_data.dropna(subset=["article_abstract", "article_date_published"])
    31  +     article_data = article_data.dropna(subset=["article_abstract", "article_date_published", "article_title"])
32  32        article_data = article_data.loc[article_data.article_type == "JOURNAL-ARTICLE"]
33  33        article_data = article_data.dropna(subset=["article_type"])
```

MDPI

Goal

Writing reliable and maintainable DataScience code.

Disclaimer: These best practices are opiniated. If I don’t mention your favorite library or tool, talk to me in the coffee break.

Content

- Type annotation
- Code setup
- Erdos distance: Classical objects
- Erdos distance: DataFrames
- What did we learn?

MDPI

Theory speed run

MDPI

Long story short

Type annotation works well for classical objects, support for Data Frames is limited.

MDPI

Type annotation ⟷ **Data validation**

```python
def factorial(n: int) -> int:
    """Compute the factorial of a non-negative integer."""
    ...
```

```python
def factorial(n):
    """Compute the factorial of a non-negative integer."""
    if not isinstance(n, int) or n < 0:
        raise TypeError("Input must be a non-negative integer.")
    ...
```

MDPI

Type annotation

What type annotation is:

- Metadata in the code
- Indication of type of a variable
- Used for static code analysis
- Used by IDE to suggest code completion
- Serves as documentation

→ **Development time**

What it does not do:

- Performance improvement (CPython)
- Runtime type checker

→ **No impact at runtime**

MDPI

Type annotation: Example

Annotation of variable

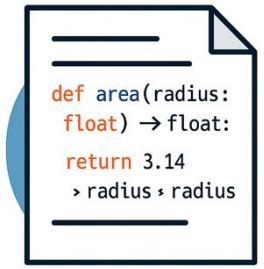
of function argument

of return value

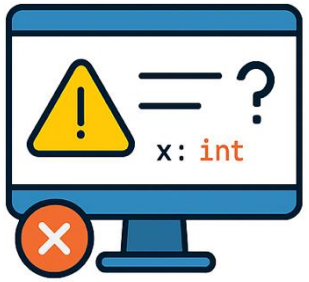
```python
MAGIC_CONST: int = 42

def add_magic_number(x: int) -> int:
    """Adds the magic constant to the given number."""
    return x + MAGIC_CONST
```

MDPI

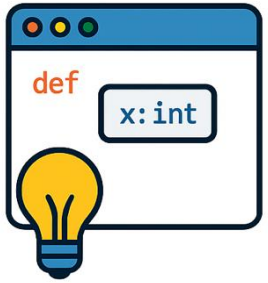
Type annotation: Benefits



Readability and maintainability



Catching mistakes early



Improved IDE suggestions

Static code analysis

We will see this in action.

Data validation

What data validation is:

- Verification object/value has certain properties at runtime
- Validation of user input or API responses
- Data validated with code
    - isinstance
    - hasattr
    - “field” in value

→ **Runtime**

What it is not:

- Duck typing
- Coding suggestions for IDEs

MDPI

Data validation: Examples

```python
def factorial(n):
    """Compute the factorial of a non-negative integer."""
    if not isinstance(n, int) or n < 0:
        raise TypeError("Input must be a non-negative integer.")
    ...
```

Library: Pydantic

Pydantic Model / “Type”

```python
from pydantic import BaseModel

class Transfer(BaseModel):
    amount: int
    currency: str


a = Transfer(amount=100, currency="EUR")
b = Transfer(amount=10.13, currency="CHF")
```

Validation

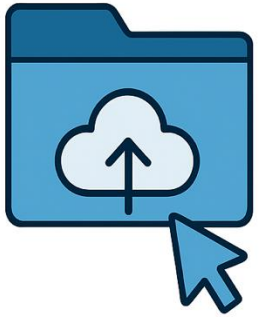
Fails once executed

MDPI

Data validation: Benefits

Guaranteed data correctness

Consistent error reporting



Safe integration with external data sources

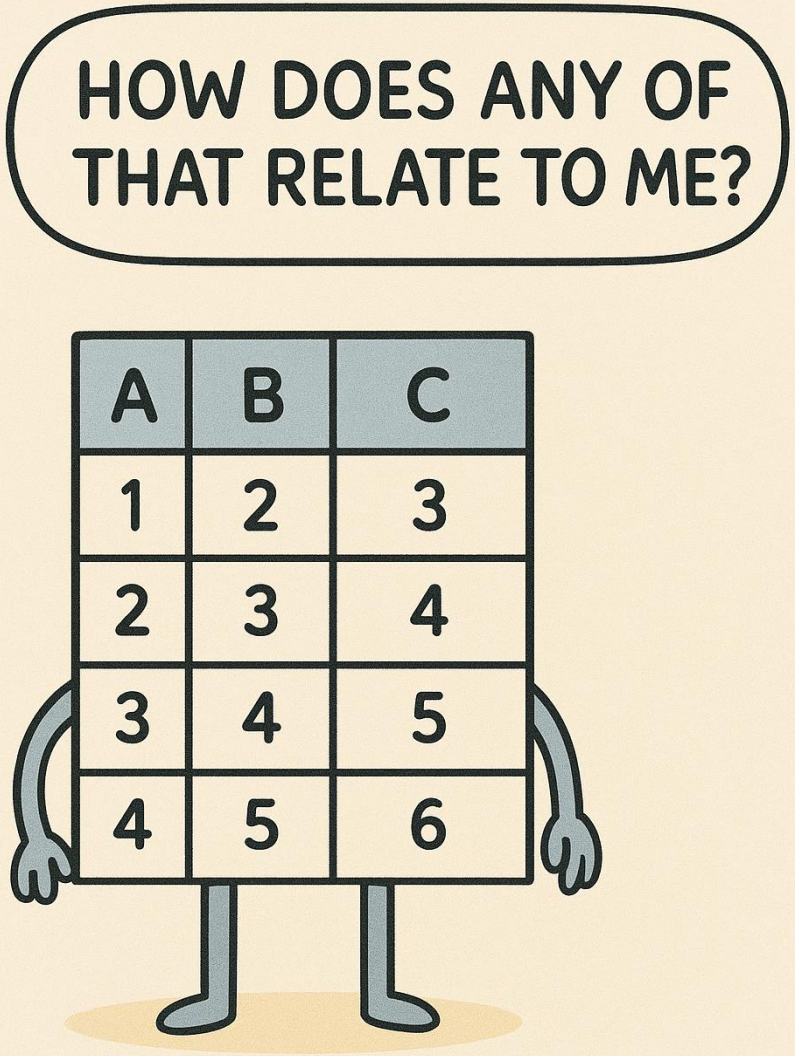
We will see this in action.

Static code analysis and IDE

- IDE VSCode → Language server Pylance
- Pylance → Pyright
- Pyright: Fast type checker

- Mypy: Extendable type checker
    - Plugins for DataFrames

MDPI



MDPI

DataFrames: Annotation vs Validation

Development time

Annotation

A DataFrame object

```
df: pd.DataFrame = ...
```

Runtime

Validation

DataFrame columns (“dynamic”)

(Read from external source)

How to annotate the “dynamic”?

MDPI

Excursion: Dict → TypeDict

- Fields defined at runtime (“dynamic”)
- IDE support for dict methods
- No support for column names
- No support for column types

```
t = {
    "amount": 100,
    "currency": "EUR"
}
```

- Fields defined at development time
- IDE support for dict methods
- IDE support for column names
- IDE support for column types
- No data validation
- Hard-coded in language servers

```
from typing import TypedDict

Transfer = TypedDict("Transfer", {
    "amount": int,
    "currency": str
})

t = Transfer(amount=100, currency="EUR")
```

MDPI

Are we there yet?

Dict

```
object: dict = ...
```

TypedDict

+ Annotation

Pydantic

+ Annotation
+ Validation

DataFrame

```
df: pd.DataFrame = ...
```

?

MDPI

Annotating DataFrame columns

- **Stub files**
  - Pandas stubs, Pandera stubs
  - Add type annotations (native) modules

→ not considered in this tutorial

- **Native Schemas**
  - pl.DataFrame(…, schema=schema)
  - Runtime validation

→ not considered in this tutorial

- **Pandera Schemas**
  - Some support of column-level annotation
  - Runtime validation

MDPI

Pandas, polars, and more

This tutorial focuses on Pandas.

Technical setup

Interactive Tutorial

- https://github.com/MDPI-AG/euroscipy2025/
- Disable Copilot, disable mypy
- Package manager: uv or requirements.txt

	Live code suggestions	Live type checks	Static type checks (mypy)	Notebooks
VSCode preferred	Yes	Yes	Yes	Yes
PyCharm	Yes	Yes	Yes	Yes?
Binder Browser			Yes	Yes
Vim / Terminal			Yes	In Browser

MDPI

Erdős number

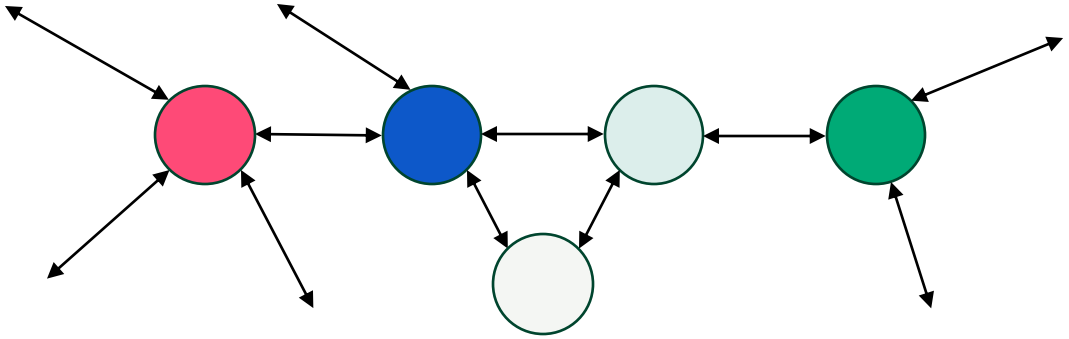
3 + 4

MDPI

- Open access scientific publisher
- Making research accessible
- 7000 staff world-wide
- Headquarter in Basel
- Office in Krakow
- Innovative AI Team, 25+ members

Dijkstra's Algorithm in the co-authorship graph

Erdős number



- Author data and article data
- Function to compute Erdős distance between to authors

Measurements of Higgs Boson …

G. Aad, F. Sauerburger, B. Abbott, et al.

Distance +1

Cite Lens: An AI Tool for Detecting …

JB. de la Broise, F. Sauerburger, et al.

Distance +2

How to pretrain an efficient …

JB. de la Broise, JP. Dubuc, et al.

Distance +3

Distance(G. Aad, JP. Dubuc) = +3

Repo and Tasks

Plain Python

TypedDict

Pydantic

DataFrames

DataFrame Schema

jupyter

jupyter

MDPI

Tasks

```python
display_paper(fat_articles[0])
```

```
(tutorial) frank.sauerburger@Frank-MDPI tutorial % python erdos_pydantic.py
Plant-Fungal Interactions: A Case Study of <i>Epicoccoum nigrum</i> Link
Authors: Agata Piecuch, Katarzyna Przywara, Agnieszka Lejman, Rafał Ogórek, Krzysztof Matkowski
```

1. **Plain Python:** Familiarize with data and code
2. Run `mypy erdosX_xxx.py` and see what it detects
3. Implement a function that displays first fat_paper:
   Plain Python, TypedDict, or Pydantic

4. **DataFrames + Pandera:** Familiarize
5. ScoreCard
6. Remove column in pandera schema

MDPI

Score card

```
authors = load_ndjson("esp2025_authors_buggy.ndjson")
```

```
attach_authors(articles, authors, authorships)
```

```
articles[0]
```

	Live Suggestions			Reporting				
	Method	Key/Column	Item/Column type	Non-existent Item/Column	Non-existent Item/Column mypy	Argument mistake	Mistake reporting mypy	Data Validation
Plain Python								
TypedDict								
Pydantic								
DataFrame								
DataFrame + Schema								

```
articles
```

```
build_coauthorship_matrix(authors, authorships)
```

MDPI

Score card

```
authors = load_ndjson("esp2025_authors_buggy.ndjson")
```

```
attach_authors(articles, authors, authorships)
```

```
articles[0]
```

	Live Suggestions			Reporting				
	Method	Key/Column	Item/Column type	Non-existent Item/Column	Non-existent Item/Column mypy	Argument mistake	Mistake reporting mypy	Data Validation
Plain Python	×	×	×	×	×	×	×	KeyError
TypedDict	Yes	Yes	Yes	×	Yes	×	Yes	KeyError
Pydantic	Yes	Yes	Yes	Yes	Yes	×	Yes	Yes
DataFrame	Yes	×	x	×	×	×	×	float64
DataFrame + Schema	Yes	×	×	×	×	×	Yes	Yes

```
articles
```

```
build_coauthorship_matrix(authors, authorships)
```

MDPI

What did we learn?

Are we there yet?

Dict

```
object: dict = ...
```

TypedDict

Pydantic

DataFrame

```
df: pd.DataFrame = ...
```

?

MDPI

Pandera schemas

- Check if correct schema used as arguments
- Doesn’t check if columns are correctly used at development time (Annotation)
- Does validate schema at runtime (Validation)
- For lots of operations (groupby) the output schema is determined,
  but it doesn’t happen with Pandera*

*It’s not Pandera’s fault, Python type annotation system, ist not flexible enough.
In rust, schema inferrence is one of the most beautiful features.

Overview

		Type Annotation*	Runtime Validation
Classical objects	**Plain dicts**		
	TypedDict	Yes	
	Pydantic	Yes	Yes
DataFrames	**Plain DataFrame**		
	Stubs	limited	
	Native Schema		Partially
	Pandera Schema	limited	Yes

MDPI

*For dataframes this means column-level annotations

Conclusion

Conclusion

Type annotation in Python → Extremely useful

Support for DataFrames is limited → Use data validation

Aware of “Annotating the dynamic”? → Great

MDPI

Thank you!

@MDPIOpenAccess

facebook.com/MDPIOpenAccessPublishing

linkedin.com/company/mdpi

MDPI AG – YouTube

instagram.com/mdpiopenaccess

www.mdpi.com

Advanced topics

Complex Types

- Simple types: int, float, str, float
- Compound types:
  - list[str] list of strings
  - tuple[str, int] a string and an integer
  - dict[int, str] mapping integers to strings
  - set[int] set of integers
- Callable[int, float] function taking integers and returning floats

MDPI

Type variables

Beautiful example: FastAPI + Pydantic

MDPI

Stub files